



# A BRIEF INTRODUCTION INTO PYTHON

|  
*Week 1 of 3*



## OVERVIEW

- Anaconda & IDEs
  - *Spyder & Jupyter Notebooks*
- Data Types in Python
- Basic Mathematic Operators & Functions
- Libraries and Packages
  - *NumPy, SciPy, Matplotlib*

# ANACONDA & IDES

# LET'S DOWNLOAD ANACONDA

---

What's Anaconda?

*Open-source distributor for  
python/R-IDEs*

What's an IDE?

Integrated Development  
Environment





- An IDE
- Best formatted for writing ".py" scripts and testing them out
- Variable Viewer and Directory Viewer

The screenshot shows the Spyder IDE interface. On the left, the **Code Editor** (highlighted with a red border) displays a Python script named `workshop.py`. The script imports various libraries and performs data exploration and visualization on a weather dataset. On the right, the **Variable Viewer/File Explorer** (highlighted with a green border) provides help for objects and shows a "New to Spyder? Read our tutorial". Below it, the **Console/Command Line** (highlighted with a yellow border) shows the Python environment and an IPython session. The bottom status bar indicates the Python version (3.9.2), LSP status, Kite indexing, memory usage (68%), and file paths.

```
# -*- coding: utf-8 -*-
# Copyright © Spyder Project Contributors
# Licensed under the terms of the MIT License
"""Workshop main flow."""

# In[1] Importing Libraries and Data
# Third-party imports
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import explained_variance_score

# Local imports
from utils import (
    plot_correlations, plot_color_gradients, aggregate_by_year,
    predicted_temperature)

# In[2] Exploring Data
# Reading data
weather_data = pd.read_csv('data/weatherHistory.csv')

# Print size of data
print(len(weather_data))

# Print first 3 rows of DataFrame
print(weather_data.head(3))

# TO DO: Print the last 3 rows of the DataFrame
print(weather_data.tail(3))

# In[3] Visualisation
# Order rows according to date
weather_data['Formatted Date'] = pd.to_datetime(
    weather_data['Formatted Date'])
weather_data_ordered = weather_data.sort_values(by='Formatted Date')

# Order Index according to date
weather_data_ordered = weather_data_ordered.reset_index(drop=True)

# Drop categorical columns
weather_data_ordered = weather_data_ordered.drop(
    columns=['Summary', 'Precip Type', 'Loud Cover', 'Daily Summary'])

# Plot Temperature Vs Formatted Date
weather_data_ordered.plot(
    x='Formatted Date', y=['Temperature (C)', color='red', figsize=(15, 8))
```

Code Editor

Console/  
Command Line

# JUPYTER NOTEBOOKS

---

- Interactive Python Editor
- Can write basic ".py" scripts and notebooks ".ipynb"
- Annotate Notebooks



WE'RE USING JUPYTER NOTEBOOK TODAY!



Home



**Anaconda Toolbox**  
Supercharged local notebooks.  
Click the Toolbox tile to Install.  
[Read the Docs](#)

Documentation

Anaconda Blog



All applications on base (root) Channels

DataSpell <p>DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.</p> <a href="#">Install</a>	Anaconda Notebooks <p>Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.</p> <a href="#">Launch</a>	Jupyter Notebook 6.5.4 <p>Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.</p> <a href="#">Launch</a>	Qt Console 5.4.2 <p>PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.</p> <a href="#">Launch</a>	Spyder 5.4.3 <p>Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features</p> <a href="#">Launch</a>
Datalore <p>Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.</p> <a href="#">Launch</a>	IBM Watson Studio Cloud <p>IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.</p> <a href="#">Launch</a>	Oracle Cloud Infrastructure <p>OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools</p> <a href="#">Launch</a>	Glueviz 1.2.4 <p>Multidimensional data visualization across files. Explore relationships within and among related datasets.</p> <a href="#">Install</a>	JupyterLab 3.6.3 <p>An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.</p> <a href="#">Install</a>
Orange 3 3.34.0 <p>Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.</p>	PyCharm Professional <p>A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.</p>	RStudio 1.1.456 <p>A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.</p>	<h2>Launch Jupyter Notebook</h2>	

WEEK 1

# Creating a new Notebook

The screenshot shows the Jupyter Notebook interface. At the top right are 'Quit' and 'Logout' buttons. Below them is a navigation bar with 'Files' (selected), 'Running', and 'Clusters' tabs. A message 'Select items to perform actions on them.' is displayed above the file list. On the right side of the toolbar are 'Upload', 'New' (with a dropdown menu), and a refresh icon. The main area is a file browser with a sidebar showing a directory tree: '0 /' at the root, followed by 'anaconda3', 'Applications', 'Creative Cloud Files', 'Desktop', 'Documents', 'Downloads', 'Dropbox (GaTech)', 'Movies', 'Music', 'Pictures', 'Public', and a file 'matlab\_crash\_dump.15791-1'. To the right of each item are columns for 'Name', 'Last Modified', and 'File size'. A red box highlights the 'New' button in the toolbar.

Name	Last Modified	File size
0 /		
anaconda3	8 months ago	
Applications	9 months ago	
Creative Cloud Files	a day ago	
Desktop	5 minutes ago	
Documents	17 days ago	
Downloads	3 hours ago	
Dropbox (GaTech)	17 days ago	
Movies	a year ago	
Music	a year ago	
Pictures	a year ago	
Public	a year ago	
matlab_crash_dump.15791-1	a month ago	10.2 kB

# Creating a new Notebook

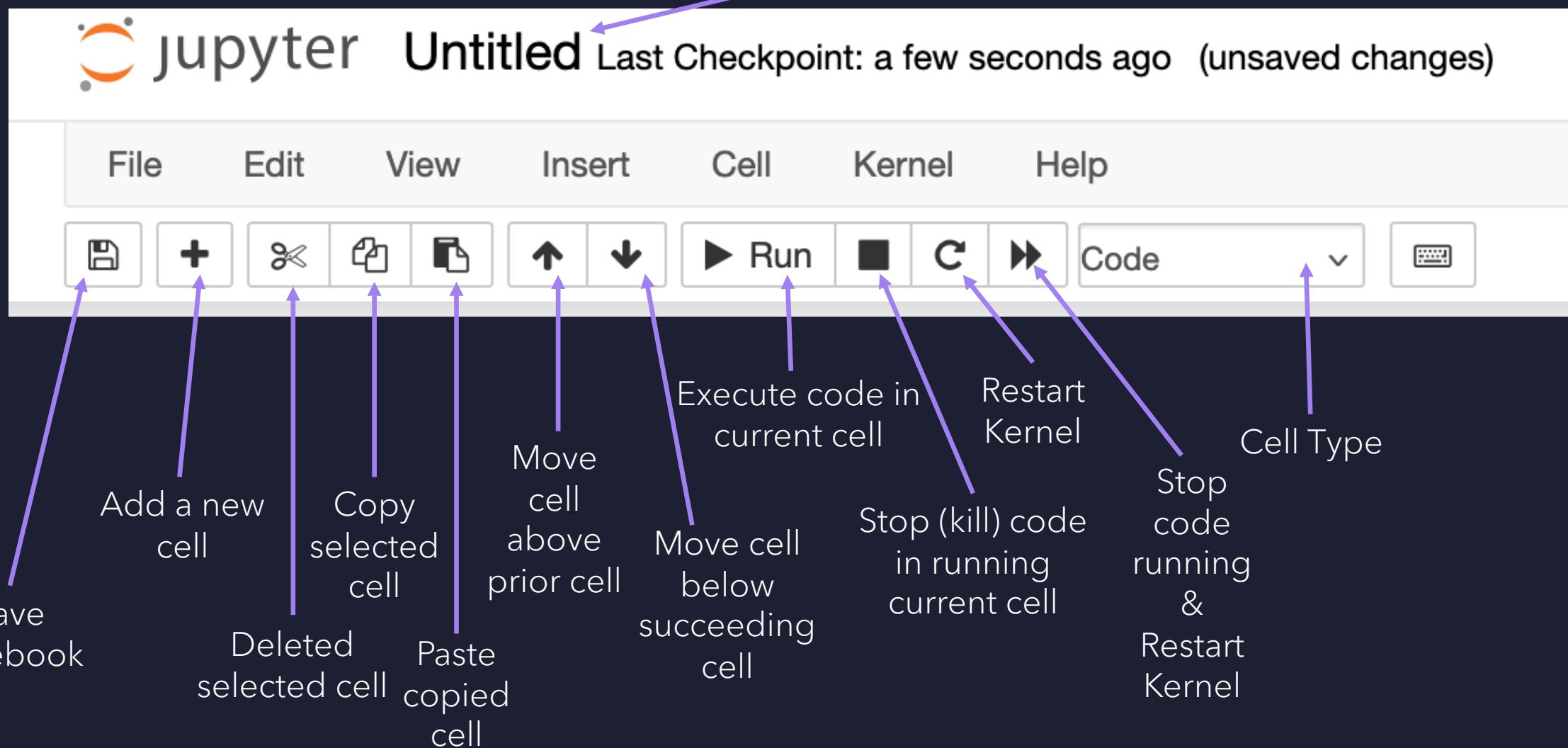
The screenshot shows the Jupyter Notebook interface. At the top, there is a navigation bar with the Jupyter logo, 'Files' (selected), 'Running', 'Clusters', 'Quit', and 'Logout'. Below the navigation bar, a message says 'Select items to perform actions on them.' On the left, there is a file tree with a dropdown menu showing '0' and a folder icon. The tree lists several directories: 'anaconda3', 'Applications', 'Creative Cloud Files', 'Desktop', 'Documents', 'Downloads', 'Dropbox (GaTech)', 'Movies', 'Music', 'Pictures', 'Public', and 'matlab\_crash\_dump.15791-1'. To the right of the file tree, there is a toolbar with 'Upload', 'New ▾' (with a dropdown menu), and a refresh icon. The 'New ▾' dropdown menu is open, showing options: 'Notebook:' (selected), 'Text File', 'Folder', and 'Terminal'. The 'Notebook:' option has a red box around it. The dropdown also includes a 'Name' field with 'Python 3 (ipykernel)' and a 'size' field. Below the dropdown, there is a list of files with their last modified times and sizes: '17 days ago', '3 hours ago', '17 days ago', 'a year ago', 'a month ago 10.2 kB'. The entire interface is set against a dark background.

# NEW NOTEBOOK IS MADE!

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** "jupyter Untitled Last Checkpoint: a few seconds ago (unsaved changes)"
- User Area:** Python 3 (ipykernel) selected in the top right.
- Toolbar:** Includes File, Edit, View, Insert, Cell, Kernel, Help, Trusted, and Logout buttons.
- Cell Buttons:** Includes icons for file, plus, asterisk, copy, paste, up, down, run, stop, clear, and execute.
- Code Selection:** A dropdown menu set to "Code".
- Input Cell:** An empty code cell labeled "In [ ]:" with a green border.

# NOTEBOOK SHORTCUT BAR BUTTONS



# SOME PYTHON VOCAB

---

- **Kernel:** Code interpreter that executes code in Jupyter notebook cell line by line
- **Console:** Command line interpreter that executes code from scripts line by line
- **Variable:** a reserved memory location to store data/values
- **Object:** A variable that contains data or functions the can be used to manipulate data
- **Element:** Items/Values inside a list, array, matrix, etc.

# DATA TYPES IN PYTHON

# DATA CATEGORIES

---

- Numeric
- String
- List
- Boolean
- Dictionary

# NUMERIC

Integer: 1, 1000, 786

Float: 0.87, 0.3333, 1.4

Complex: 8j, 5+ 2j, 80.0j

# STRING

Declared using “double quotations” or ‘single quotations’:

“Hello World”

“1+2”

‘There are 8 cookies’

# LIST

- An ordered collection of values
- 1-D horizontal vector/array
- Denoted with square brackets []

Examples:

[0, "eleven", 0.4, 8]

[1, 5, 9, 13]

# BOOLEAN

- Logical Data Type
- Only two possible outputs: **True** or **False**

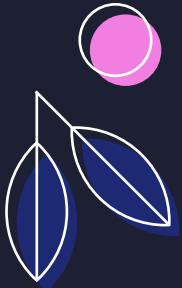
# DICTIONARY

- Unordered collection of Data Values
- Uses Curly Brackets {}
- Syntax: {"Key": Value}

# BASIC PYTHON OPERATORS & FUNCTIONS

# DECLARING A VARIABLE

- Python is object-orientated programming language
- Variables are such an object
- Variables store data
- Syntax: **varname = data**
- Naming Rules:
  - No Spaces (Use an `_` instead!)
  - Variable Name cannot start with a number
  - No special symbols



# BUILT-IN FUNCTIONS

- `len()`: Returns the length of the variable
- `type()`: Returns the data-type of a variable
- `print()`: Returns the value of a variable
- `float()`: Returns variable as a float  
(input types: integer or string only)
- `int()`: Returns variable as an integer
- `str()`: Returns variable as a string
- `bool()`: returns variables as a Boolean
  - Returns `False` when fed 0
  - Returns `True` for everything else

## MATHEMATICAL OPERATORS

Addition: +

Subtraction: -

Multiplication: \*

Division: /

Exponent: \*\*

## LOGICAL OPERATORS

Greater than: >

Less than: >

Greater than or equal to: >=

Less than or equal to: <=

Equal to: ==

# OTHER FUNCTIONS

- Comment: Placing a `#` in a line causes what followed to be commented out
- Cells: `#%%` creates a new cell (not applicable for Jupyter notebooks)
- Comment Blocks: Placing `'''` above and below a block of code comments out the entire block
- `max()`: Returns the element with the largest value in a list
- `min()`: Returns the element with the smallest value in a list

# PYTHON PACKAGES AND LIBRARIES

# LIBRARIES & PACKAGES

- Libraries and Packages are external code you load to simplify your own script
- At the beginning of every script, you should load any libraries and packages you need
- Most used: NumPy, Matplotlib, & SciPy
- Built into Anaconda (Full list [here](#))

## Syntax

```
import package_name as shortcut_name
```

Ex.: import numpy as np

Let's try loading these in our notebook!

# ALWAYS IMPORT PACKAGES AND LIBRARIES FIRST

First part of writing a python script is to import the necessary packages and libraries you will be using in your script.



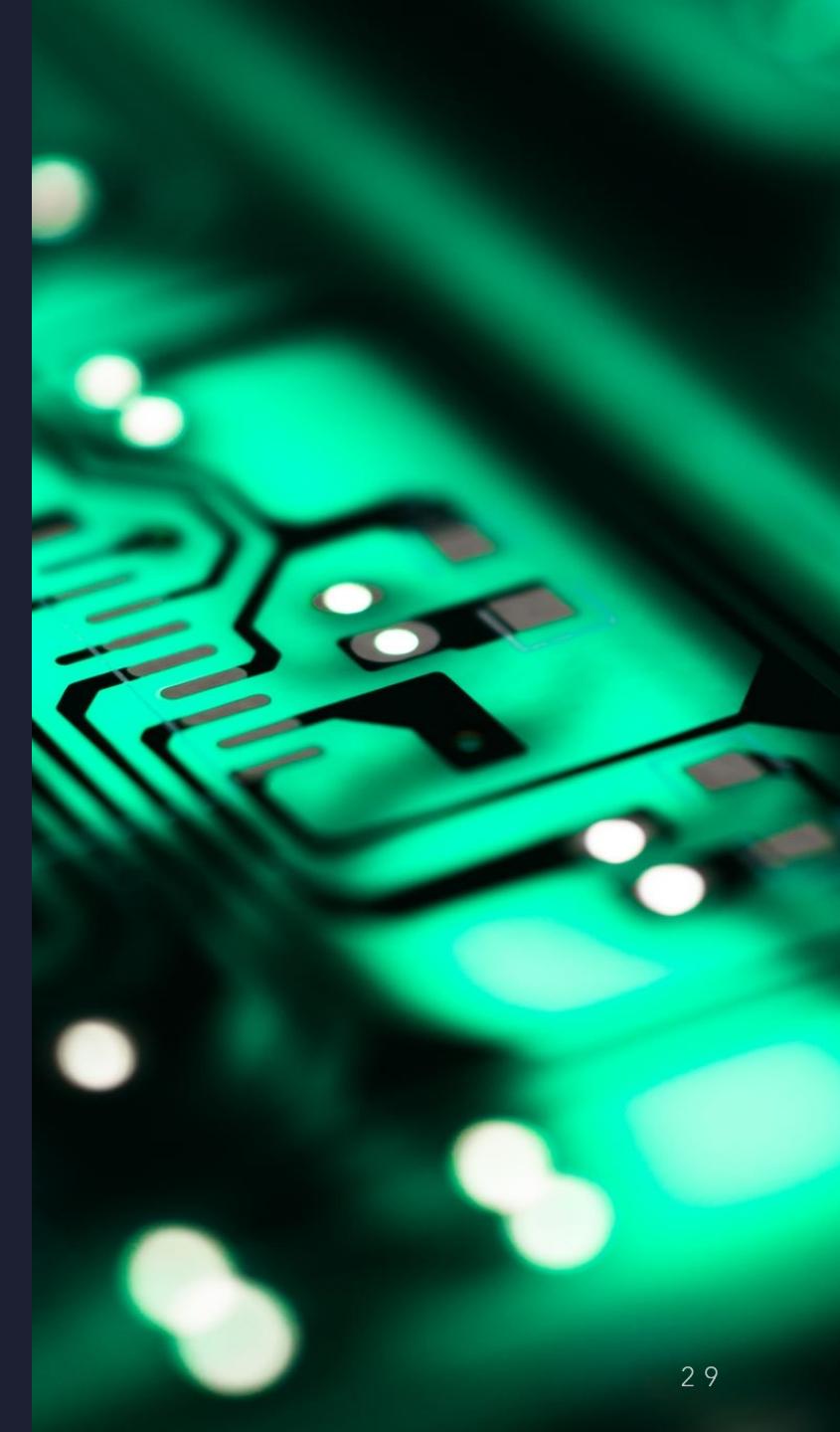


# NumPy

---

- Numerical Computing Library
- Introduces new data type "numpy array"
- Arrays are the better list
  - They can be multidimensional
  - Stores objects or "nested" sequences

Syntax: `np.array([data])`



## OTHER NUMPY ARRAY FUNCTIONS

---

`np.ones(shape)`: returns an array of ones of given shape/size

---

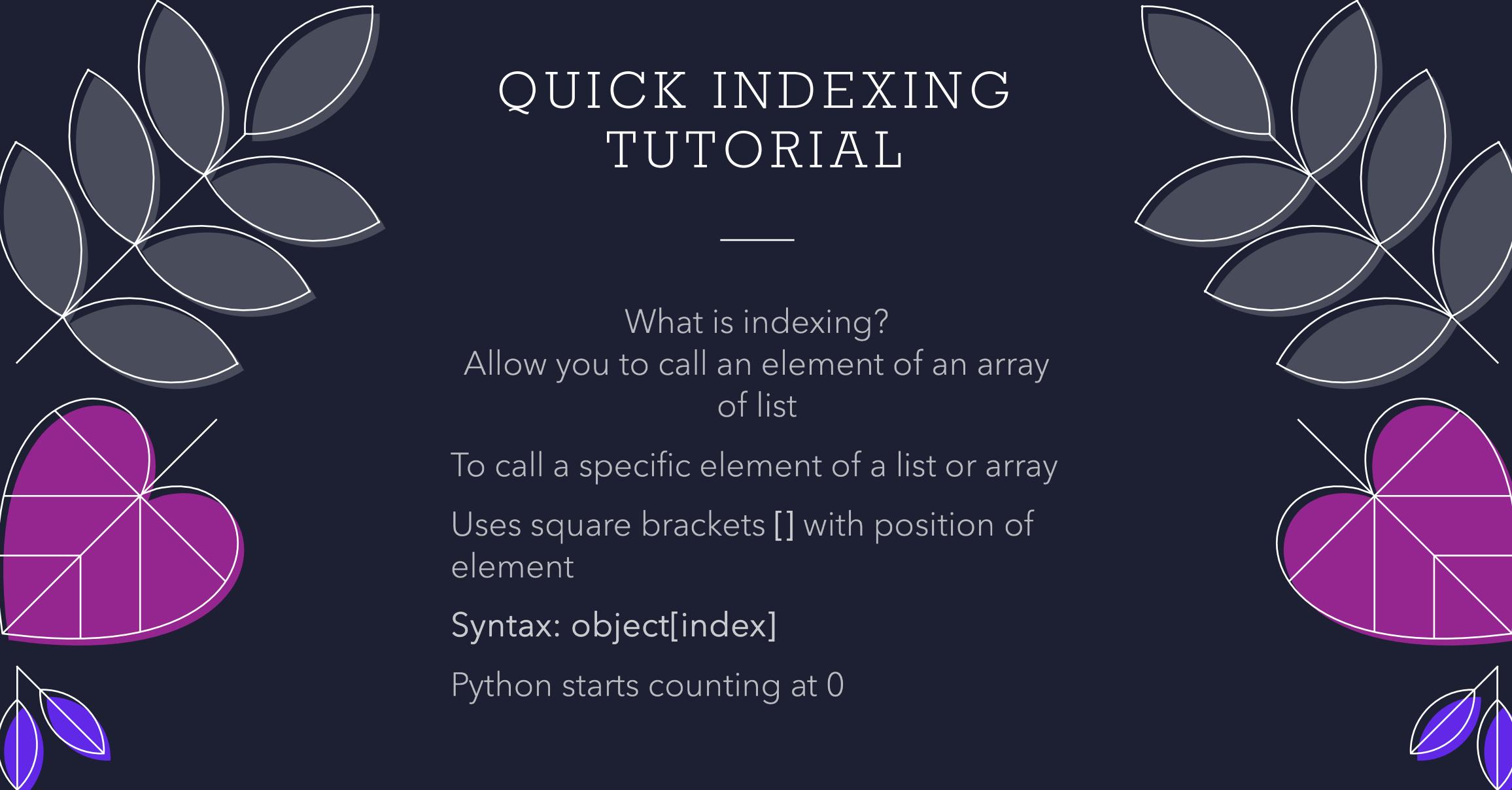
`np.zeros(shape)`: returns an array of zeros of given shape/size

---

`np.linspace(start,stop,length)`: returns array of size `length` from `start` to `stop`

---

`np.arange(start,stop,step)`: returns array from `start` to `stop` with `step` interval size between each element



# QUICK INDEXING TUTORIAL

---

What is indexing?

Allow you to call an element of an array  
of list

To call a specific element of a list or array

Uses square brackets [] with position of  
element

Syntax: object[index]

Python starts counting at 0



## INDEXING SYNTAX

- Calling a single element:  
`variable[element position]`
  - Calling several element values:  
`variable[start:stop:step]`
    - If you leave step blank it assumes `step=1`
- Element position must be a **WHOLE** number
- Using a negative number reverses the order of list or array
- Starts from the end -> start



# INDEXING STRINGS

---

- You can index strings!
- It counts the characters within the string like elements of a list or array
- You can “splice” a string and it will return a string with only those characters

Example:

Var = "Hello World"

Var[1:5] -> "ello"

Note: Splice will cutoff at element before stated end point

# MATHEMATICAL FUNCTIONS OF NUMPY

# SIMPLE MATH FUNCTIONS OF NUMPY

---

`np.round( input, decimal places)`: rounds input to the decimal place indication

---

`np.sum(input)`: sums all the elements in an object

---

`np.sqrt(input)`: takes the square root of an object

---

`np.exp(input)`: calculates the exponential ( $e^{input}$ ) of an object

---

`np.log(input)`: take the natural logarithm ( $\ln$ ) of the input

---

`np.log10(input)`: takes the logarithm with base 10 ( $\log_{10}$ ) of the input

---

`np.log2(input)`: takes the logarithm with base 2 ( $\log_2$ ) of the input

---

`np.abs(input)`: takes the absolute value of the input

# TRIGONOMETRIC FUNCTIONS OF NUMPY

---

`np.degrees(input)`: converts input from radians to degrees  
Identical function: `np.rad2deg()`

---

`np.radians(input)`: converts input from degrees to radians  
Identical function: `np.deg2rad()`

---

`np.sin(input)`: takes the sine of the input  
`np.arcsin(input)`: takes the inverse sine of the input

---

`np.cos(input)`: takes the cosine of the input  
`np.arccos(input)`: takes the inverse cosine of the input

---

`np.tan(input)`: takes the tangent of the input  
`np.arctan(input)`: takes the inverse tangent of the input

# STATISTICAL FUNCTIONS OF NUMPY

**np.max(input):**  
returns the maximum value of the input object

**np.min(input):** returns the minimum value of the input object

**np.median(input):**  
returns the median of an object

**np.mean(input):**  
returns the mean of an object

**np.average(input):**  
returns the weighted average of an object

**np.std(input):** returns the standard deviation of an object

**np.var(input):** returns the variance of an object

**np.histogram(input):**  
returns the histogram of an object

# MATHEMATICAL CONSTANTS IN NUMPY

# MATHEMATICAL CONSTANTS IN NUMPY

---

- Pi ( $\pi$ ): `np.pi`
- Infinity ( $\infty$ ): `np.inf`
- Negative infinity ( $-\infty$ ): `np.ninf`
- Euler's constant ( $e$ ): `np.e`
- Not a Number (nan): `np.nan`

# matplotlib



- Comprehensive library for creating static, animated, and interactive visualizations in Python
- It's the most used plot maker in Python
- Other plotting packages and libraries typically have some dependency on Matplotlib
- Pyplot is most common package

Import Matplotlib as:

```
import matplotlib.pyplot as plt
```



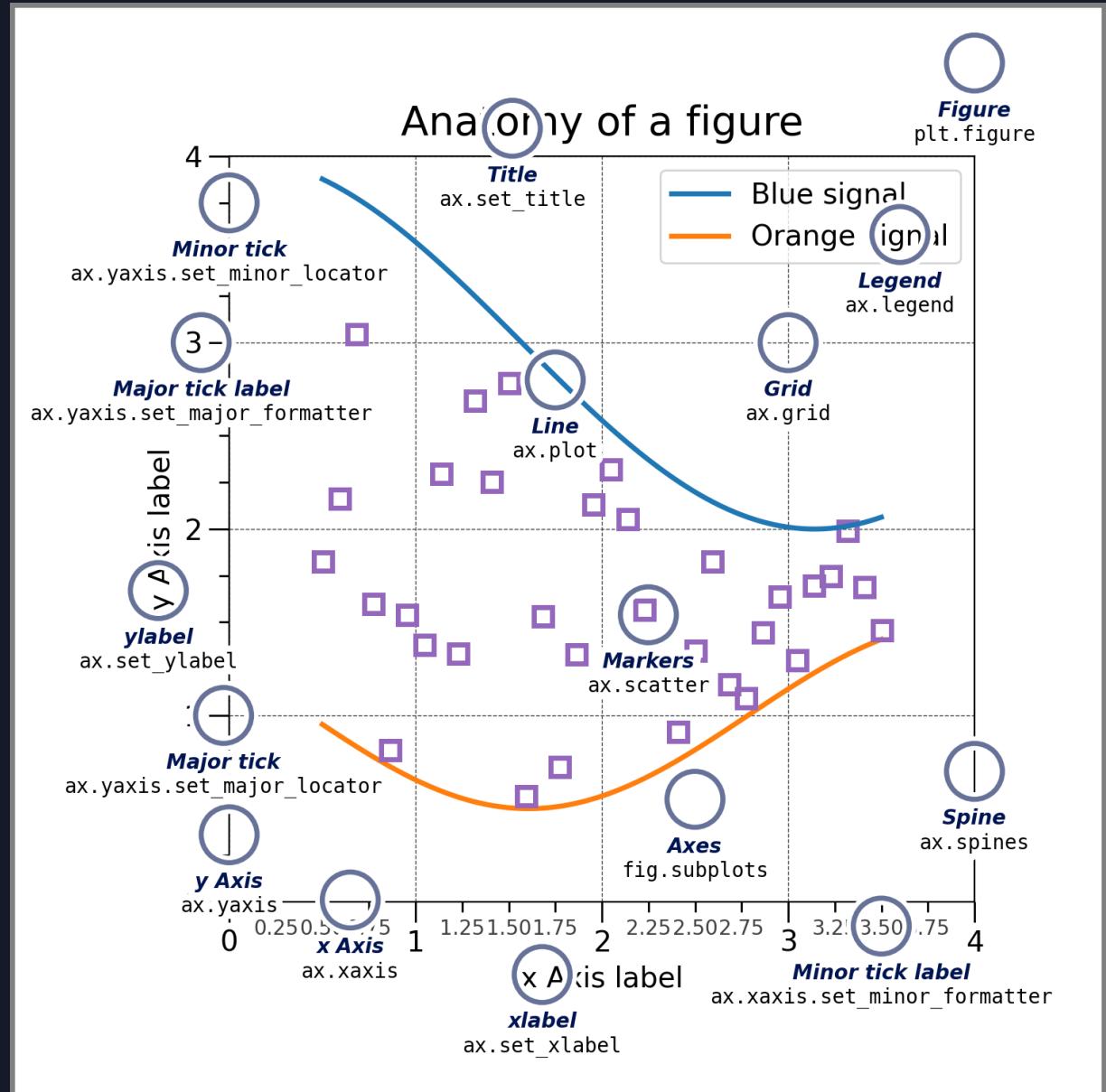
# GETTING STARTED

- Make sure your lists/arrays are the **same length** (have the same number of elements)
  - Always begin a new plot with creating a new Figure

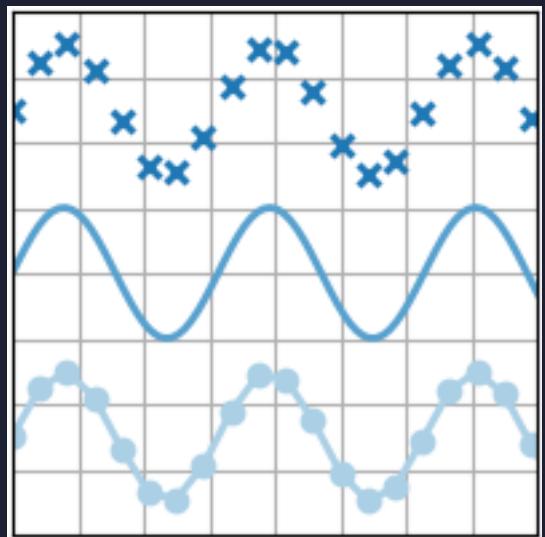
**Figure:** blank window which plots/ subplots are displayed on

- You overlap plots in a single **Figure**
  - You can put multiple plots in a single figure with **subplots**

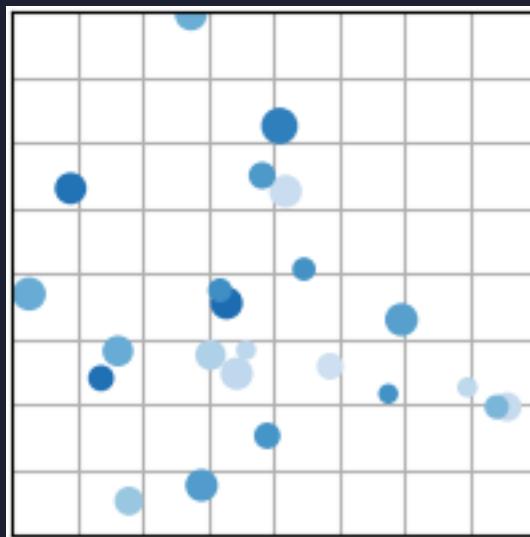
To create a new figure use plt.figure()



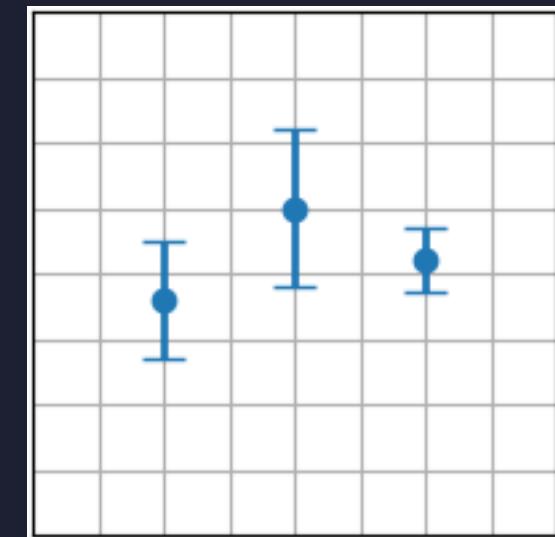
# COMMON 2D PLOTS



```
plt.plot(x,y)
```

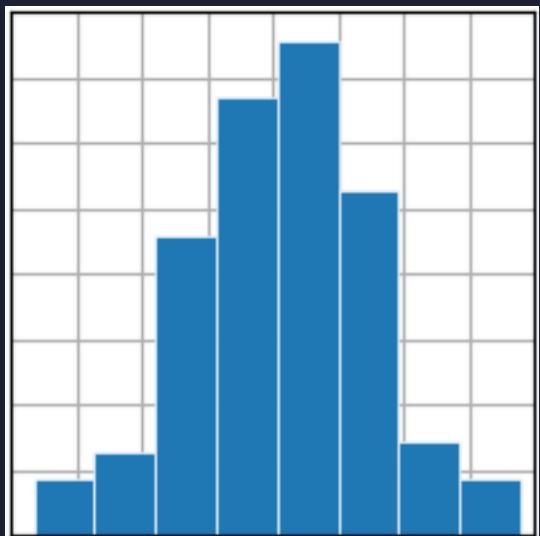


```
plt.scatter(x,y)
```

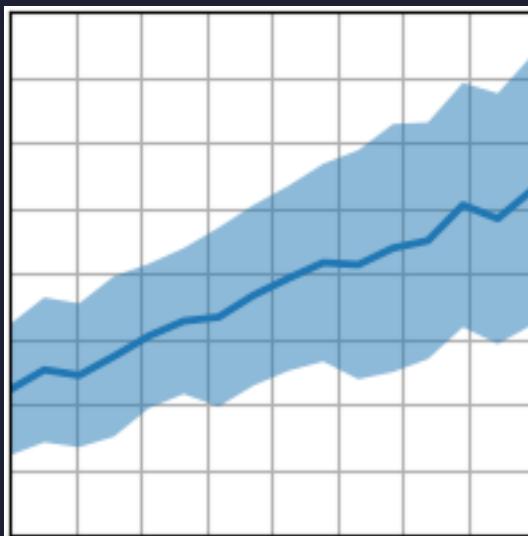


```
plt.errorbar(x,y,xerror,yerror)
```

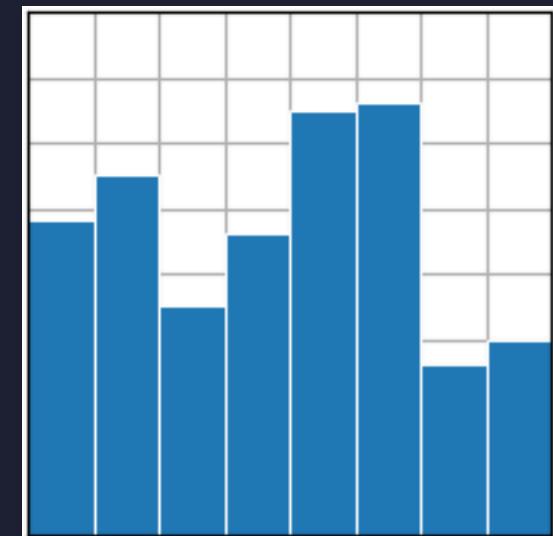
# OTHER IMPORTANT PLOT TYPES



`plt.hist(x)`

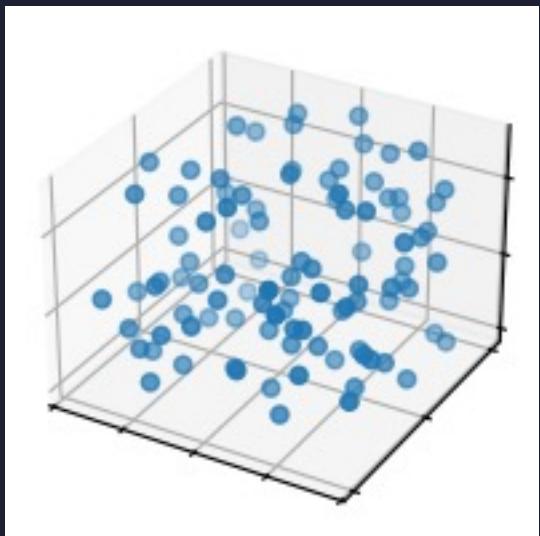


`plt.fill_between(x,y1,y2)`

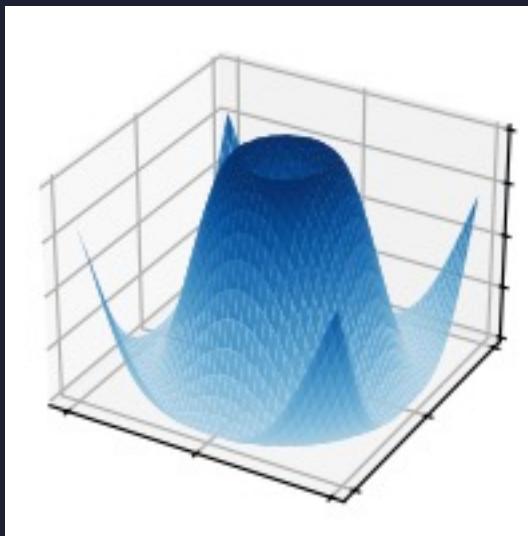


`plt.bar(x,y)`

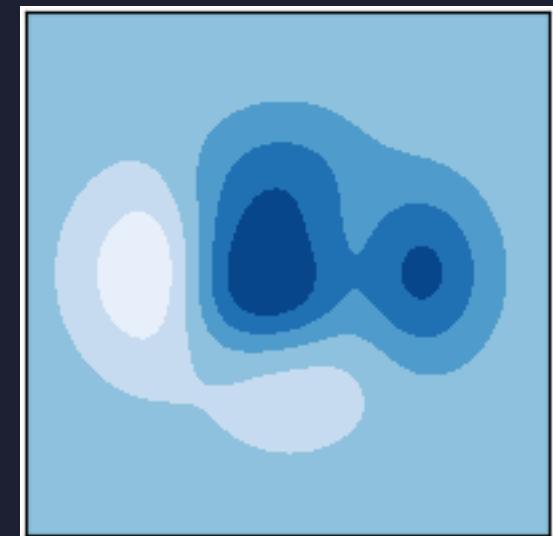
# SOME 3D DIMENSIONAL PLOTS



`plt.scatter(x,y,z)`



`plt.plot_surf(x,y,z)`



`plt.contourf(x,y,z)`



## CUSTOMIZE PLOTS

---

- There are a wide arrange of colors and markers you can use to customize your plot
- You can also add labels to the vertical and horizontal axes  
X-axis: `plt.xlabel("String Input")` & Y-axis: `plt.ylabel("String Input")`
- You can title your plot with the following function: `plt.title("String Input")`

All label and titles require a string as a minimum input

There are customizations for font, font size, super/subscripts, symbols, etc.

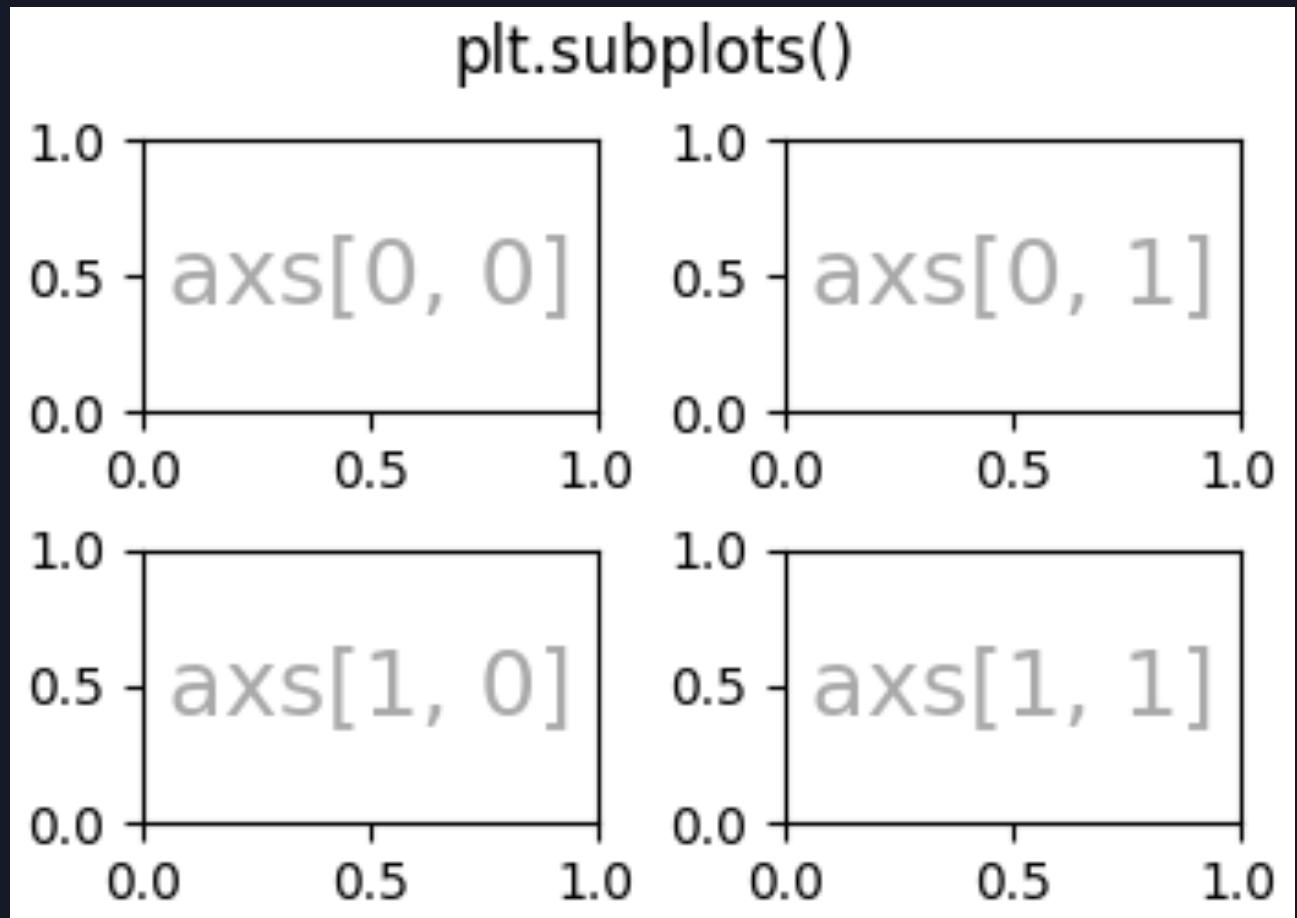
For multiple sets of data in a single plot window, you can add a label within the specific plot function you call (Ex. `plt.plot(x,y,label="string")`) and place a legend (`plt.legend()`) on the plot

LET'S TRY CREATING A PLOT IN JUPYTER  
NOTEBOOK

# SUBPLOTS

---

- You can add multiple plot windows in the same figure using `plt.subplots(# rows, #columns)`
- The function gives two outputs a **figure** and **axes** object
  - it can be a singular variable or an array of variable for each subplot



# LET'S MAKE A FIGURE OF SUBPLOTS IN JUPYTER NOTEBOOKS

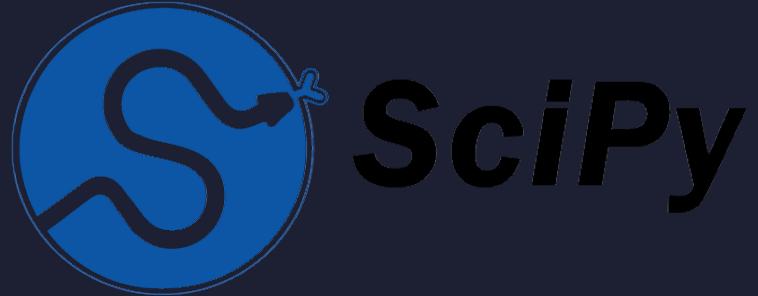
# CUSTOMIZING SUBPLOTS

---

- Customizations can be made to each individual plot in a subplot figure
- For each axes variable output from `plt.subplots`, you are able perform the same customizations as the `plt.plot`
- Syntax: `axes_var.set_customization`

Let's try a few in Jupyter Notebook





- Library for scientific computing built from NumPy
- Used primarily to solve mathematical and scientific problems
- Capable of performing integration, linear algebra, solving ODEs, signal processing, etc.

We'll learn more about SciPy during Week 3 in some Data Processing

WEEK 1



A GOOD CHUNK OF  
CODING IS GOOGLING.  
GOOGLE AND STACK  
OVERFLOW ARE YOUR  
BEST FRIENDS



# SUPPLEMENTAL

# KEYBOARD SHORTCUTS

---

- Shift+Enter: to run
- Control+/: Comment/Uncomment current line
- Package/library Shortcut.Tab: to see list of functions in a package
- Tab: Autocompletes variable names and function names

